Contenido

Pasos DABD para el parcial ANGULAR

Pasos:

-“ng new” ponemos eso mas el nombre del proyecto y se crea el proyecto. Damos a si al rotuing.

-“ng g class models/” + nombre del modelo, creamos los modelos necesario para el enuciado el modelo que contenga fk se crea Ej: una variable idCategoria : number;

- en el tsconfig.json agregamos lo siguiente en el compilerOptions = “stritctPropertyInitialization”: false,

- en el index.html pegamos los links de Bootstrap:

```
<!-- CSS only -->
<link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.2/dist/css/bootstrap.min.css"
 rel="stylesheet"
 integrity="sha384-
Zenh87qX5JnK2Jl0vWa8Ck2rdkQ2Bzep5IDxbcnCeuOxjzrPF/et3URy9Bv1WTRi"
crossorigin="anonymous">
<!-- JavaScript Bundle with Popper -->
<script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.2/dist/js/bootstrap.bundle.min.js"
integrity="sha384-
OERcA2EqjJCMA+/3y+gxIOqMEjwtxJY7qPCqsdltbNJuaOe923+mo//f6V8Qbsw3"
crossorigin="anonymous"></script>
```

Servicios

-“ng g service services/” + el nombre del servicio EJ: articulo o Categoria
-Creamos una variable string para la URL y una lista de tipo arreglo en este caso un Categoria.

categoria.service.ts

```
import { Injectable } from '@angular/core';
import { Categoria } from '../models/categoria';
import { HttpClient} from '@angular/common/http';
import { Observable } from 'rxjs';

@Injectable()
export class CategoriaService {
  API_URL: string=
"https://634892bb0b382d796c73ec11.mockapi.io/Categoria/";

  private listado : Categoria[];
  constructor( private http : HttpClient) { this.listado=[]; }

  obtenerCategoria() : Observable<Categoria[]>{
    return this.http.get<Categoria[]>(this.API_URL);
  }
}
```

articulo.service.ts

```
import { Injectable } from '@angular/core';
import { HttpClient} from '@angular/common/http';
import { Articulo } from '../models/articulo';
import { Observable} from 'rxjs';

@Injectable() //eliminar el provider root este y despues ir al app
// module  y dentro del provider poner nombre del serivicio
export class ArticuloService {
   private API_URL : string =
"https://634892bb0b382d796c73ec11.mockapi.io/Categoria/1/Articulos/";
  private listado : Articulo[];

  constructor(private http : HttpClient) {this.listado=[]; }
//inicializamos el arreglo

  agregar(articulo:Articulo) : Observable<Articulo>{
    return this.http.post<Articulo> (this.API_URL,articulo);
  }
  eliminar(articulo:Articulo) : Observable <Articulo>{
    return this.http.delete<Articulo> (this.API_URL+articulo.id);
  }
  obtenerListado() : Observable<Articulo[]>{
    return this.http.get<Articulo[]>(this.API_URL);
  }
  modificar(articulo : Articulo): Observable<Articulo>{
    return this.http.put<Articulo> (this.API_URL+articulo.id,articulo);
  }
  getArticulo(id: number) : Observable<Articulo>{
```

```
    return this.http.get<Articulo>(this.API_URL+id);
  }
}
```

app.module.ts

```
  imports: [
    BrowserModule,
    AppRoutingModule,
    HttpClientModule, //agregar esto y el FromsModule
    FormsModule
  ],
  providers: [ArticuloService,CategoriaService],
  //agregar estos servicios
```

Routing

app.component.html

-Borramos todo y agregamos un navbar el cual vamos a usar con routing, al navbar lo copiamos de Bootstrap y borramos lo que no necesitemos.

```
<nav class="navbar navbar-expand-lg bg-light">
  <div class="container-fluid">
    <button class="navbar-toggler" type="button" data-bs-
toggle="collapse"
    data-bs-target="#navbarSupportedContent" aria-
controls="navbarSupportedContent"
    aria-expanded="false" aria-label="Toggle navigation">
      <span class="navbar-toggler-icon"></span>
    </button>
    <div class="collapse navbar-collapse" id="navbarSupportedContent">
      <ul class="navbar-nav me-auto mb-2 mb-lg-0">
        <li class="nav-item">
          <a class="nav-link"  href="#" routerLinkActive="active"
          routerLink="home">Inicio</a>
        </li>
        <li class="nav-item">
          <a class="nav-link" href="#" routerLinkActive="active"
          routerLink="alta-articulo">Alta articulo</a>
        </li>
        <li class="nav-item">
          <a class="nav-link" href="#" routerLinkActive="active"
          routerLink="lista-articulo">Listado articulo</a>
        </li>

       </ul>
    </div>
  </div>
</nav>
```

```
<div class="container-fluid">
  <router-outlet></router-outlet>
  <!-- renderiza las paginas de los componentes que vamos a crear -->
</div> ESTO ES MUY IMPORTANTE VA EN EL ROUTING HTML
```

app-routing.module.ts

```
const routes: Routes = [
  {path : 'home',component: HomeComponent},
  {path : 'alta-articulo', component: AltaArticuloComponent},
  {path : 'lista-articulo',component: ListaArticuloComponent},
  {path : 'alta-articulo/:id',component: AltaArticuloComponent},
  {path : '', redirectTo: 'home',pathMatch:'full'}
]; // estas son las rutas que creamos, tiene que coincidir con el router
link del navbar para redireccionar
```

Creamos Componentes

-Se crea con el comando "ng g c "+ el nombre del componente
-Creamos un componentes header y home

header.componente.ts

Agregamos en la clase:

```
  @Input() title ?: string;
  @Input() subtitle ?: string;
```

header.component.html

```
<div class="container-fluid bg-primary py-5">
    <div class="container text-white">
        <h1 class="display-3">{{title}}</h1>
        <p class="lead">{{subtitle}}</p>
    </div>
</div>
```

Home.component.html

```
<app-header [title]="'Pagína Principal'"
[subtitle]="'Bienvenido!'"></app-header>
```

Luego creamos los componentes para realizar el ABM Componentes: Alta,Baja y Modificar, además de los componentes que interactúen con el abm: contenedor-articulo y lista-articulo.

Buscamos en Bootstrap un formulario y lo modificamos a gusto ejemplo:

-Componente alta

alta-articulo.component.html

```
<app-header [title]="!isEdit ? 'Alta artículo' : 'Editar Artículo'"
[subtitle]="'Ingrese los datos'"></app-header>
<div class="container">
    <form #articuloForm="ngForm">
      <!-- hacemos esto ngForm si o si para validar antes de apretar
guardar o etc -->
        <div class="mb-3">
```

```
          <label for="inputCodigo" class="form-label">Codigo</label>
          <input type="text" class="form-control" id="inputCodigo" aria-
describedby="inputCodigo" name="codigo"
          [(ngModel)]="articulo.codigo"
          #codigo="ngModel"
          required
          [class.is-invalid]="codigo.touched&&codigo.invalid"
          >
          <div *ngIf="codigo.touched&&codigo.invalid" class="alert alert-
danger">
            El codigo es requerido,complete el campo.
          </div>
        </div>
        <div class="mb-3">
            <label for="inputNombre" class="form-label">Nombre</label>
            <input type="text" class="form-control" id="inputNombre"
aria-describedby="inputNombre" name="nombre"
            [(ngModel)]="articulo.nombre"
            #nombre="ngModel"
            [class.is-invalid]="nombre.touched&&nombre.invalid"
            required
            >
            <div *ngIf="nombre.touched&&nombre.invalid" class="alert
alert-danger">
                El nombre es requerido,complete el campo.
              </div>
          </div>
          <div class="mb-3">
            <label for="inputPrecio" class="form-label">Precio</label>
            <input type="text" class="form-control" id="inputPrecio"
aria-describedby="inputPrecio" name="precio"
            [(ngModel)]="articulo.precio"
            #precio="ngModel"
            [class.is-invalid]="precio.touched&&precio.invalid"
            required
            >
            <div *ngIf="precio.touched&&precio.invalid" class="alert
alert-danger">
                El precio es requerido,complete el campo.
              </div>
          </div>
          <div class="mb-3">
            <label for="inputStock" class="form-label">Stock</label>
            <input type="text" class="form-control" id="inputStock" aria-
describedby="inputStock" name="stock"
            [(ngModel)]="articulo.stock"
            #stock="ngModel"
            [class.is-invalid]="stock.touched&&stock.invalid"
            required
```

```
            >
            <div *ngIf="stock.touched&&stock.invalid" class="alert alert-
danger">
                El stock es requerido,complete el campo.
            </div>
          </div>
          <div class="mb-3">
            <label for="inputCategoria" class="form-
label">Categoria</label>
            <select id="inputCategoria" class="form-select"
name="categoria"
            [(ngModel)]="articulo.idCategoria"
            #categoria = "ngModel"
            >
              <option *ngFor="let categ of listadoCategorias"
[value]="categ.id">
                {{categ.nombre}}
              </option>
            </select>
          </div>
        <button type="submit" class="btn btn-outline-primary"
*ngIf="!isEdit" (click)="guardar()">Guardar</button>
        <button type="submit" class="btn btn-outline-success"
*ngIf="isEdit" (click)="editar()">Editar</button>
        <button type="button" class="btn btn-outline-danger"
routerLinkActive="actived" routerLink="/home">Cancelar</button>
      </form>
</div>
```

alta-articulo.component.ts

```
import { Component, OnInit, ViewChild,Output,EventEmitter } from
'@angular/core';
import { ActivatedRoute, Router } from '@angular/router';
import { ArticuloService } from '../services/articulo.service';
import { CategoriaService } from '../services/categoria.service';
import {NgForm} from '@angular/forms';
import { Subscription } from 'rxjs';
import { Articulo } from '../models/articulo';
import { Categoria } from '../models/categoria';

@Component({
  selector: 'app-alta-articulo',
  templateUrl: './alta-articulo.component.html',
  styleUrls: ['./alta-articulo.component.css']
})
export class AltaArticuloComponent implements OnInit {

  articulo: Articulo;
  listadoCategorias : Categoria[];
```

```
  isEdit: boolean = false;

  @ViewChild('articuloForm') formulario : NgForm;
  @Output() onConfirmar = new EventEmitter();
  private subscription= new Subscription();
  constructor(private servicioArticulo : ArticuloService,
    private activatedRoute: ActivatedRoute,
    private router: Router,
    private servicioCategoria: CategoriaService) { }

  ngOnInit(): void {
    this.articulo= new Articulo();
    this.obtenerCategorias();
    this.cargar();
  }
  ngOnDestroy(): void {
    this.subscription.unsubscribe();
  }
  //los siguientes metodos copiamos tal cual en todos los proyectos,
modificamos la variable
  guardar(){
    this.subscription.add(
      this.servicioArticulo.agregar(this.articulo).subscribe({
        next : () =>{
          this.onConfirmar.emit();
          alert('Registro el artículo correctamente');
          this.router.navigate(['lista-articulo']);
        },
        error : () =>{
          alert('Error al guardar artículo');
        }
      })
    )
  }
  obtenerCategorias(){
    this.subscription.add(
      this.servicioCategoria.obtenerCategoria().subscribe({
        next : (listado : Categoria[]) =>{
          this.listadoCategorias=listado;
        },
        error : () =>{
          alert('Error al obtener categorias');
        }
      })
    )
  }
  editar(){
    this.servicioArticulo.modificar(this.articulo).subscribe({
      next : () =>{
```

```
        alert('Edito el artículo correctamente');
        this.router.navigate(['lista-articulo']);
      },
      error : () =>{
        alert ('Error al editar');
      }
    })
  }
  cargar () : void{
    this.activatedRoute.params.subscribe(
      e=>{
        let id = e['id'];
        if(id){
          this.isEdit=true;
          this.servicioArticulo.getArticulo(id).subscribe(
            es => this.articulo=es
        )}else{
          this.isEdit=false;
        }
      }
    )
  }
}
```

-Componente Listado

Buscamos en Bootstrap un listado y lo modificamos

lista-articulo.component-html

```
<app-header [title]="'Listado de artículos'" [subtitle]="'Informacíon de
artículos'"></app-header>
<div class="container">
    <table class="table">
        <thead>
          <tr>
            <th scope="col">ID</th>
            <th scope="col">Codigo</th>
            <th scope="col">Nombre</th>
            <th scope="col">Precio</th>
            <th scope="col">Stock</th>
            <th scope="col">Categoria</th>
            <th scope="col">Eliminar</th>
            <th scope="col">Editar</th>
          </tr>
        </thead>
        <tbody>
          <tr *ngFor="let articulo of listado;">
            <th scope="row">{{articulo.id}}</th>
            <td>{{articulo.codigo}}</td>
```

```
            <td>{{articulo.nombre}}</td>
            <td>{{articulo.precio}}</td>
            <td>{{articulo.stock}}</td>
            <td>{{articulo.idCategoria}}</td>
            <!-- cargamos con esto el contenido de la tabla y con
            interpolacion le seteamos los valores -->
            <td>
                <app-baja-articulo [articulo]="articulo"
(onEliminado)="actualizarListado()"></app-baja-articulo>
            </td>
            <td>
              <app-editar-articulo [articulo]="articulo"></app-editar-
articulo>
            </td>
          </tr>
        </tbody>
      </table>
</div>
```

Lista-articulo.component.ts

```
@Component({
  selector: 'app-lista-articulo',
  templateUrl: './lista-articulo.component.html',
  styleUrls: ['./lista-articulo.component.css']
})
export class ListaArticuloComponent implements OnInit {
  @Input() listado : Articulo[];
  constructor(private servicioArticulo : ArticuloService) { }
  private subscription = new Subscription();
  ngOnInit(): void {
    this.actualizarListado();
  }
  actualizarListado(){
    this.subscription.add(
      this.servicioArticulo.obtenerListado().subscribe({
        next : (listado : Articulo[]) =>{
          this.listado=listado;
        },
        error : () =>{
          alert('Error al actualizar listado');
        }
      })
    )
  }
}
```

-Componente baja

baja-articulo.component.html

```
<button class="btn-close" aria-label="close"
(click)="eliminarArticulo()"></button>
```

baja-articulo.component.html

```
export class BajaArticuloComponent implements OnDestroy {
  @Input() articulo : Articulo;
  @Output() onEliminado = new EventEmitter();
  constructor(private servicioArticulo : ArticuloService) { }
  //importante hacer uso del servicio
  private subscription = new Subscription();
  ngOnDestroy() : void {
    this.subscription.unsubscribe();
  }
  eliminarArticulo(){
    this.subscription.add(
      this.servicioArticulo.eliminar(this.articulo).subscribe({
        next : () =>{
          this.onEliminado.emit();
          alert('Elimino el articulo con id'+' '+ this.articulo.id+' '
+'correctamente');
        },
        error : () =>{
          alert('Error al eliminar');
        }
      })
    )
  }
}
```

-Componente editar

editar-articulo.component.html

```
<button class="btn btn-success" routerLinkActive="active"
[routerLink]="['/alta-articulo',articulo.id]">Editar</button>
```

editar-articulo.component-ts

-agregamos el input en la clase

```
  @Input() articulo : Articulo;
```

CONTENEDOR

-Componente contenedor

Es el padre de los componentes de artículos

contenedor-articulo.component.html

```
<app-alta-articulo (onConfirmar)="actualizarListado()"></app-alta-
articulo>
<app-lista-articulo [listado]="listadoArticulos"></app-lista-articulo>
```

contenedor-articulo.component.ts

```
import { Component, OnDestroy} from '@angular/core';
import { ArticuloService } from 'src/app/services/articulo.service';
import { Subscription } from 'rxjs';
import { Articulo } from 'src/app/models/articulo';
@Component({
  selector: 'app-contenedor-articulo',
  templateUrl: './contenedor-articulo.component.html',
  styleUrls: ['./contenedor-articulo.component.css']
})
export class ContenedorArticuloComponent implements OnDestroy {
   listadoArticulos : Articulo[];
  constructor(private servicioArticulo : ArticuloService) { }
  private subscription = new Subscription();
  ngOnDestroy(): void {
    this.subscription.unsubscribe();
  }

  actualizarListado(){
    this.subscription.add(
      this.servicioArticulo.obtenerListado().subscribe({
        next : (listado : Articulo[]) =>{
          this.listadoArticulos=listado;
        },
        error : () =>{
          alert('Error al actualizar listado');
        }
      })
    )
  }

}
```